There are two sections, each worth 50 points. Partial credit is possible for each question. The maximum score for the test is 100 points.

It is essential that you **PUT YOUR NAME ON ALL TEST MATERIALS**. It can be difficult to identify the author of an unsigned test and it would be better to avoid this problem.

**Section 1**: Below you will find several Pieces of Code followed by a question and a place to fill in an answer. Assume that there are no bugs in the code that will make the system crash, although some code may not solve a particular problem perfectly. If you find anything that you think is a bug, there is either a typo (and I should fix it for everyone) or you are mistaken.

**Sample Question A**

```
output = '1'+'1'
```

Question: What does output equal?
Answer: '11'
Note: Attention to detail is important. The quotes indicate that it is a string. Partial credit is possible. For example, leaving out the quotes would have lost just a little bit, but answering 2, would have resulted in an incorrect answer.

**Question 1**

```
def character_count(string,target_character):
    count = 0
    for found_character in string:
        if target_character == found_character:
            count = count+1
    return(count)
output = character_count('Mississippi','s')
```

Question: What does output equal after the above function call?
Answer:
def convert_to_morse_code(digit):
    if digit == '0':
        return('_____')
    elif digit == '1':
        return('.____')
    elif digit == '2':
        return('..___')
    elif digit == '3':
        return('...__')
    elif digit == '4':
        return('...._')
    elif digit == '5':
        return('.....')
    elif digit == '6':
        return('_....')
    elif digit == '7':
        return('__...')
    elif digit == '8':
        return('___..')
    elif digit == '9':
        return('____.')
    else:
        print('Illegal Digit: ', digit)

def digits_in_morse_code(number):
    output = ''
    for digit in str(number):
        if output == '':
            output = convert_to_morse_code(digit)
        else:
            output = output + '|' + convert_to_morse_code(digit)
    return(output)

output = (digits_in_morse_code(2011))

Question: What does output equal after the above function call?
Answer:
Question 3:

```python
import turtle
my_screen = turtle.Screen()
turtle1 = turtle.Turtle()
turtle2 = turtle.Turtle()

def move_both_turtles(distance):
    turtle1.fd(distance)
turtle2.fd(distance)

def turn_turtles_opposite(degrees):
    turtle1.left(degrees)
turtle2.left(degrees+180)

def turtle_infinity():
    turtle1.pd()
turtle2.pd()
    turn_turtles_opposite(90)
    ## The circle method takes one parameter: the radius of the circle
    ## the circle method creates a circle tangent
    ## to the direction it is facing.
    ## For example, if it is facing rightward, the circle will
    ## appear on top of the turtle’s current position.
    turtle1.circle(40)
turtle2.circle(40)
turtle1.pu()
turtle2.pu()

turtle_infinity()
```

Question: Draw the shape (approximately) that the turtles would draw.

Answer:
def breeding_rabbits(generations):
    ## this function simulates the population growth in rabbits
    ## when there are no predators
    total1 = 0
    total2 = 0
    for number in range(1, generations + 1):
        if total1 == 0:
            total1 = 1
            ## We buy one pair of baby rabbits
        elif total2 == 0:
            total2 = 1
            ## After one month they mature and are ready to mate
        elif total1 <= total2:
            total1 = total1 + total2
            ## Afterwards there are always some rabbits that are too
            ## young to reproduce and all the other rabbits do reproduce.
            ## Thus rather than doubling, you simply add the 2 previous totals.
            ## The lower total is the number that can reproduce and the higher total
            ## is the total number of rabbits (including the immature ones).
        else:
            total2 = total1 + total2
            if total1 > total2:
                return total1
            else:
                return total2

output = breeding_rabbits(10)

Question 4a: What does output equal after the above function call?
Answer 4a:

Question 4b: What were the final values of total1 and total2, when the final result was returned?
Answer 4b:
Section 2: Write functions as specified.

Question 5: Write a function that uses the following global variables to count occurrences of the corresponding types of characters:

- vowels = 'aeiou'
- consonants = 'bcdfghjklmnprstvwxyz'
- digits = '0123456789'

The program should do the following:

1. Take a string as a parameter.
2. Initialize four counters to zero. There should be one counter for each of the following types of characters: vowels, consonants, digits, and other.
3. Loop through the string, one character at a time and increment the counters depending on the type of character. Use the global variables above to determine the type of character.
4. Print the results in an informative way.
5. Return the total number of characters.

Given the input string: 'I want to finish this test in 5 minutes and leave.'
your program should print the following (or equivalent text):
vowels: 15 consonants: 23 digits: 1 other: 11
and the program should return the number: 50
Question 6: Write a program that will ask a user several questions about how nice they are. The program will then calculate how nice they are in Standard Niceness Units (SNUs), according to guidelines devised by the World Council on Niceness (WCN). These guidelines are represented in the flowchart below. The flowchart indicates when questions should be asked and how many SNUs are added or subtracted to the total SNU score based on the answers. The questions are given as global variables. You should use these variables instead of writing out all the questions (it will save a lot of time). At your option, you may also use (without writing it yourself), the function `yes_or_no`. You can vary from the flowchart if it doesn’t matter logically. For example, you could ask the user questions 1, 2, 3, 4, 5 and 6 and use the answers (stored as variables) in the if/else statements of your decision tree if this seems easier to you.

The `yes_or_no` function and global variables:

```python
def yes_or_no(question):
    answer = ''
    while not (answer in ['Yes','yes','No','no']):
        print('Please answer the following question "yes" or "no". ')
        answer = input(question)
        if answer in ['Yes','yes']:
            return(True)
        else:
            return(False)
```

```python
question1 = 'Do you always share your candy? '  
question2 = 'Do you clean up after yourself most of the time? ' 
question3 = 'Do you give money to charity? '  
question4 = 'Do you spend some of your time doing charitable work? '  
question5 = 'Do you care about non-human animals? '  
question6 = 'Would you commit a crime against another person for personal gain? ' 
question1a = 'Do you always share your food? '  
question2a = 'Do you clean up after other people? '  
question3a = 'Would you give up all your money to help a friend in serious need? '  
question3a1 = 'Would you do the same for a stranger? '  
question4a = 'Would you dedicate your life to a charitable cause and be poor yourself? '  
question5a = 'Do you believe in eating all non-human animals? '  
question5a1 = 'Do you believe in eating some non-human animals? '  
question6a = 'Would you steal so much that your victim becomes poor? '  
question6b = 'Would you murder another person for profit? ' 
```

Some hints for following the flow chart: questions 1,2,3,4,5,6 are mandatory. Regardless of whether question 1 is answered with 'yes' or 'no', question 2 must be asked. Regardless of whether question 2 is answered 'yes' or 'no', question 3 must be asked, etc. However, other questions are only asked under certain circumstances. In most cases a positive answer yields more questions, but not always. Usually, positive answers cause points to be added to the SNU (total Standard Niceness Units), but occasionally, positive answers cause points to be subtracted.

The flow chart is on the following page.

*For extra credit*, you can complicate the decision tree further by adding five additional questions. Only add questions that would be asked if previous questions are answered No.